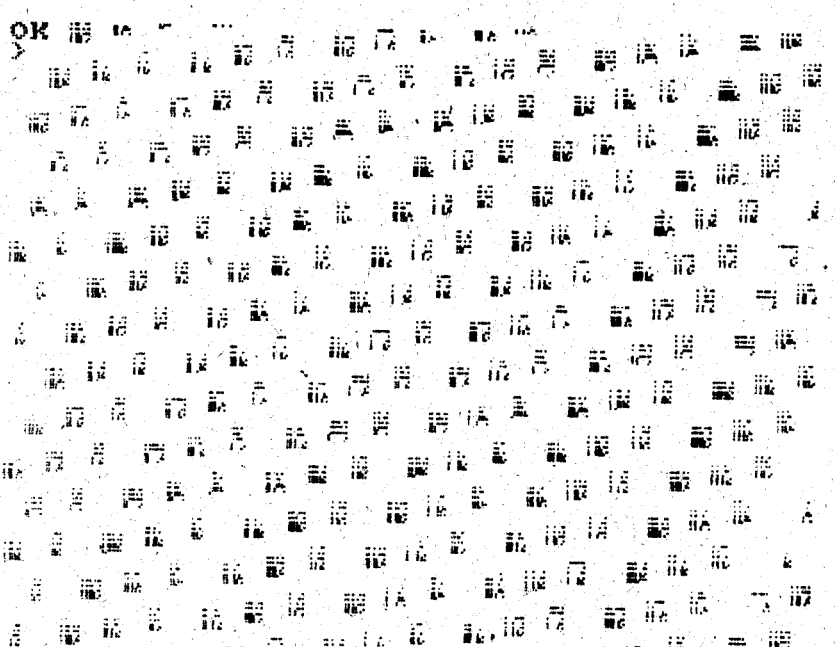
COMPUTE RE MAGAZIN

Einführung in Maschinencode

Autor:
Prof. Dr. Horst Völz

D RADIO R

```
10 WINDOW 0,31,0,39:CLS
20 FOR I=A TO 10239 STEP 119
30 A=A+1 : IF A > 255 THEN A=0
40 VPOKE I,A
50 NEXT: IF A<>9 GOTO 20
```



●●● Sie empfangen uns mittwochs um 17.oo Uhr im SCHULFUNK von Radio DDR II und am Sonnabend ab 15.15 Uhr im Computerclub von Jugendradio DT 64

Sendetag			
Schulfunk	DT 64	Thema	Inhalt
8.3.	11.3.	1	Hilfsmittel und Methoden
22.3.	25.3.	2	Hardware und Befehlstypen
5.4.	8.4.	3	Die ersten Befehle (Lade-, Jump-Austauschbefehle)
19.4.	22.4.	4	Die Arithmetik, shift und rotate
3.5.	6.5.	5	Relativsprünge und Bitbefehle
17.5.	20.5.	6	Weitere Befehle (In, Out, Interrupt, Halt ...)
31.5.	3.6.	7	Zusammenfassung - Assembler, debugging

✱ Hinweise und Anregungen zu unseren Sendungen, Lob und Kritik, Angebote und Wünsche von Computerprogrammen nehmen wir dankend entgegen!
Unsere Anschrift:

REM - das Computermagazin
Schulfunk, Radio DDR
Nalepastr.
BERLIN
1 1 6 0

Sendung 1

Dies ist kein Lehrgang für Maschinenprogrammierung. Dazu müßte ein Assembler verwendet werden. Er ist aber nicht allgemein verfügbar. Deshalb werden hier die Grundprinzipien mit der direkten Codierung über Tabellen behandelt. Dies ist natürlich nur für kleinste Programme möglich. Dennoch dürfte so das Prinzip der Maschinenprogrammierung deutlich werden.
Um die Wirkung von Programmen zu erkennen sind zumindest eine Tastatur- und eine Bildschirmroutine notwendig. Sie sind für jeden Rechner unterschiedlich. Deshalb wird für diesen Kurs auf das algemein verfügbare BASIC zurückgegriffen. Dies steht im Widerspruch zu üblichen Entwicklungsumgebungen. Auch als Werkzeug (tool) wird hier BASIC verwendet.
Die aus den Tabellen entnommenen HEX-Code werden in DATA-Zeilen abgelegt und sind dort auch leicht editierbar. Ihr Ende wird durch das Sonderzeichen "EN" gekennzeichnet. Mit einer BASIC-Routine werden sie ab 3000H abgelegt. Damit dieser Speicherbereich verfügbar ist, beginnen die Programme immer mit CLEAR 100, 12280. Das Programm heißt DATHEX.

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## DATHEX ##": PRINT
20 FOR X=12288 TO 16388: READ A$
30 IF A$="EN" THEN X=16388: NEXT: END
40   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
50   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
60   POKE X,A*16+B
70 NEXT
80 DATA 47,48,5F,60,48,49,33,00,FF,EN: ! freizufällig gewählt
90 !12288D=3000H...16388=3FFFH, nutzbarer Speicherbereich; EN=Ende-Marke
100 !HEX-Zeichen aus DATA-Zeilen werden ab 3000H abgelegt.
```

In Zeile 20 liest READ ein Zeichen der DATA-Zeile, Zeile 40 bzw. 50 trennen den linken bzw. rechten Teil ab und wandeln ihn in eine Dezimalzahl um. Sie wird in Zeile 60 wieder zusammengefaßt und an die richtige Stelle gepokt.
Mit dieser Methode können Sie auf jedem Rechner, der die folgenden Voraussetzungen erfüllt diesen Kurs mitmachen:

- Er muß als Mikroprozessor einen U 880 bzw. Z 80 besitzen,
- Er muß über einen BASIC-Interpreter verfügen,
- Sie müssen wenige rechnerspezifische Daten kennen.

Neben dem BASIC-Text müssen Sie sich auch zuweilen den Maschinencode anschauen. Dies ist wiederum bei jedem Rechner unterschiedlich.

Für die KC 85/2 bis 4 gilt:
1) BASIC verlassen mit BYE
2) DISPLAY 3000 XXXX, die Werte für X hängen vom Beispiel ab
3) REBASIC

Für den KC 85/1 und KC 87 gilt
1) * Zusatzmonitor ZM 70 vor Beginn laden, ein Zusatz-RAM
2) BASIC verlassen mit BYE
3) ZM aufrufen
4) D3000_XXXX
5) Mit B Zusatzmonitor verlassen
6) WBASIC zurück in BASIC

Für alle anderen Rechner existieren ähnliche Hilfsmittel.

Neben dem HEX-Code ist es zuweilen auch wichtig, ASCII-Zeichen im Maschinenprogramm zu besitzen. Sie können in analoger Weise mit dem Programm DATASI abgelegt werden. Bei den KC 85/2 bis 4 können sie in der oben beschriebenen Weise angeschaut werden. Beim KC 85/1 bzw. KC 87 ist statt des o.g. Aufrufes mit D dann T3000_XXXX notwendig.

```
10 CLEAR 100, 12288: CLS: PRINT TAB(15);"## DATASI ##": PRINT
20 FOR X=12288 TO 16388: READ A$
30   IF A$="EN" THEN X=16388: NEXT: END
40   POKE X,ASC(A$)
50 NEXT
60 DATA A,A,b,F,f,2,3,X,x,EN
70 !12288D=3000H: EN=Ende-Marke
80 !ASCII-Zeichen in DATA-Zeilen werden ab 3000H abgelegt.
```

Der Aufruf der Maschinenprogramme erfolgt auf zwei Wegen:

- Sind keine Werte zu übergeben, so genügt CALL*3000 (ev.andere Adresse)
- Mit Y=USR(X) wird der Wert von X an das Maschinenprogramm übergeben und der Wert Y bringt einen Wert aus dem Maschinenprogramm zurück. Es wird zuweilen auch nur eine der beiden übergaben benötigt.

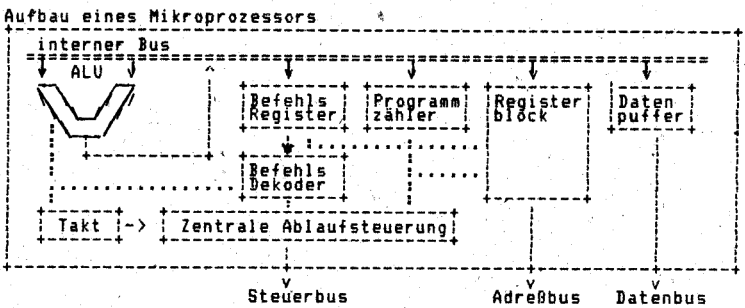
Damit die USR-Routine arbeitet, sind folgende Voraussetzungen zu erfüllen:

- Die Startadresse der Maschinenroutine muß mit DOKE 772, 12288 festgelegt werden. Eventuell ist die Adresse 12288D=3000H zu ändern. Bei anderen Rechnern (nicht KC-Typen) ist auch für 772 ein anderer Wert notwendig.
- Der Wert X ist im Maschinenprogramm durch die BASIC-Routine EIN auf das Registerpaar DE übertragbar: C96F
- Der Wert der Register A und B ist im Maschinenprogramm durch die Routine AUS auf Y übertragbar: D0B1. Die Adressen für EIN und AUS gelten wieder nur für die KC-Rechner.

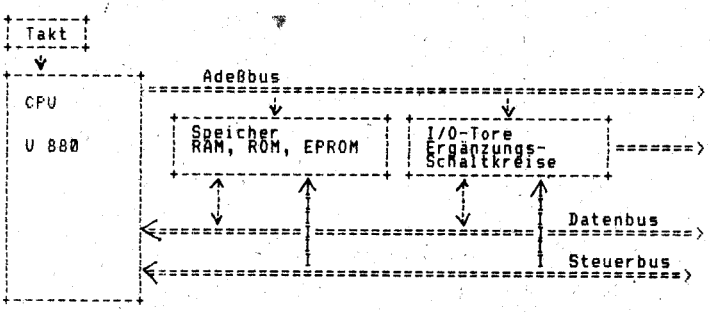
Eine Aufagabe für Sie wäre es, ein BASIC-Programm zu schreiben, mit dem Sie die abgelegten Werte ab 3000H zurückholen und anzeigen. Verwenden Sie dabei PEEK.

Sendung 2

Für die Maschinenprogrammierung ist Kenntnis über den verwendeten Mikroprozessor notwendig. Für Details sollten Sie sich Literatur beschaffen. Das folgende Bild zeigt den prinzipiellen Aufbau von Mirkroprozessoren:

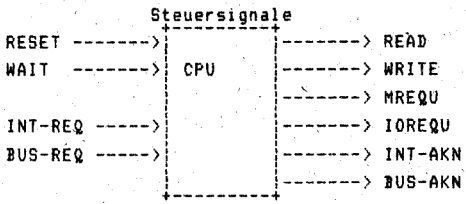


Die Arithmetisch-logische Einheit (ALU, unit) bearbeitet die Daten, die sie über den internen Bus erhält. Das dabei entstehende Ergebnis hängt vom gerade aktuellen Befehl ab. Er wird mit dem Programmzähler aus dem Speicher geholt. Dann wird über Befehlsregister, Befehlsdekoder und zentrale Ablaufsteuerung (gepunktete Linien) damit die ALU beeinflußt. Es gelangt dann wieder zurück auf den internen Bus. In dem Registerblock, der stark vom jeweiligen Mikroprozessor abhängt, werden die verschiedenen Zwischenergebnisse gespeichert.
Den Mikroprozessor verlassen drei Bussysteme: Auf dem Datenbus werden Daten von und zum Mikroprozessor übertragen. Der Adreßbus gibt zu verschiedenen Bausteinen und Geräten die Adressen aus. Der Steuerbus legt fest, was, zu geschehen hat.
Die Kopplung von Mikroprozesser mit weiteren Bausteinen zeigt das Bild:

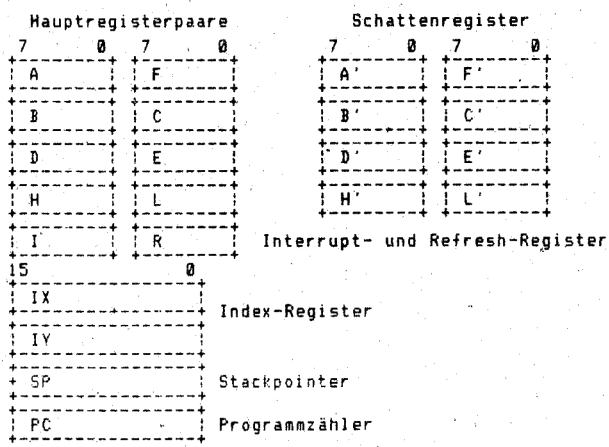


Die Erweiterung der Speicherkapazität erfolgt über ROM, RAM und EPROM. Die Werte auf dem Adreßbus legen fest, von oder zu welchen Speichern Daten auf dem Datenbus übertragen werden. Die Richtung wird durch den Steuerbus bestimmt: zu den Speichern WRITE=schreiben zur CPU READ= lesen. Auch die

anderen Schaltkreise und peripheren Geräte sind so über Adressen erreichbar. Die Unterscheidung erfolgt hier durch die Steuersignale MREQU (Memory = Speicher gefragt) bzw. IOREQU (In/Out = Ein-Ausgabe gefragt). Das Bild zeigt die Steuersignale des U 880/ Z 80. Die bisher erwähnten sind hier durch weitere hier nicht zu erklärende ergänzt.



Der Registersatz des U 880/ Z 80 ist relativ groß. Er enthält 8 Haupt- und 8 Schattenregister zu 8 Bit, zwischen denen leicht zu wechseln ist. Das Flag-Register F bzw F' besitzt eine spezielle Eigenschaft (Tab. S. 10). Es gibt Auskunft darüber wie sich das Ergebnis der ALU zeigt. Die Registerpaare BC, DE, HL können auch als 16-Bit-Register genutzt werden.



I ist das Interrupt- und R das refresh-Register. Beide werden nicht weiter besprochen. Schließlich existieren noch zwei allgemein verwendbare Index-Register, der Stackpointer und der Programmzähler zu je 16 Bit.
Die Befehle des Mikroprozessors beziehen sich auf alle bisher besprochenen Einheiten. Sie sind in den anliegenden Tabellen zusammengefaßt und werden durch ein oder mehrere HEX-Zahlen beschrieben, auch wenn es eigentlich Bitkombinationen aus jeweils acht Bit sind. Da hiermit der Mensch nicht gut umgehen kann, existieren für ihn sogenannte Mnemoniks. Dies sei an einem Beispiel erklärt:

```
verbal:     Es soll das Register A mit der Zahl 36H geladen (LD) werden
Mnemonik:   LD  A,36H                        v Assemblieren    Tab. S. 1...6
Bytefolge:  3E 36                            ^ Disassemblieren Tab. S. 6...9
Bitfolge:   00111110000110110.
```

Der Zusammenhang zwischen "verbal" und "Mnemonik" muß wie die Befehle bei BASIC erlernt werden. Zwischen den Mnemoniks und der Bytefolge vermitteln Programme wie Assembler bzw. rückwärts Dis- oder Reassembler; zwischen Byte- und Bitfolge die Betriebssysteme. Wegen der beiden Richtungen sind die Tabellen auch in zwei Gruppen zu teilen. Da wir über BASIC arbeiten, benötigen wir Umrechnungen zwischen dezimalen und Hexadezimalen Zahlen. Für ein Byte liegt die Tabelle auf S. 12 vor. Insbesondere für Adressen (für VPOKE bzw. VPEEK des KC 85/2 bis 4) gilt das Programm DEZHEX.

```
1000 CLS: PRINT TAB(15);"## DEZHEX ##": PRINT: C=65536: GOTO 1180
1010 INPUT"HEX =";A$: A=0: IF LEN(A$)>4 THEN GOSUB 1160: GOTO 1010
1020 FOR I=1 TO LEN(A$)
1030 B=ASC(MID$(A$,I,1))-48: IF B>9 THEN B=B-7:IF B<10 THEN GOSUB 1160:
     GOTO 1010
1040 IF B<0 OR B>15 THEN GOSUB 1160: RETURN
1050   A=A*16+B
1060 NEXT: PRINT A,: IF A+A>C THEN PRINT A-C,A-C/2: ELSE PRINT
1070 GOTO 1010
1080 !---------------- DEZ->HEX ----------------------
1090 INPUT"DEZ =";A: D=C: A$="": IF A<0 THEN A=A+C
1100 IF A>C OR A<0 THEN GOSUB 1160: RETURN
1110 FOR I=1 TO 4: D=D/16
1120 B=INT(A/D): A=A-B*D: B=B+48: IF B>57 THEN B=B+7
1130 A$=A$+CHR$(B)
1140 NEXT: PRINT TAB(20);A$: GOTO 1090
1150 !---------------- FEHLER ------------------------
1160 PRINT"$$ Fehler $$": RETURN
1170 !---------------- MAIN --------------------------
1180 INPUT"HEX oder DEZ =";B$: B$=LEFT$(B$,1): IF B$="H" THEN GOSUB 1010
1190 IF B$="D" THEN GOSUB 1090
1200 IF B$<>"H" AND B$<>"D" THEN GOSUB 1160: END
1210 GOTO 1180
```

Es wurde bewußt auf eine so hohe Zeilenzahl gelegt, damit Sie es immer nutzbar hinter alle Programme laden können. Das vorangehende Programm sollte dann mit **999 END** abschließen.
Das Programm BITFLAG ermöglicht Ihnen einen besseren Umgang mit den Zahlen, die im Register F zu stehen haben.

```
10 CLS: PRINT TAB(15);"## BITFLAG ##": PRINT: GOTO 30
20 PRINT "## Fehler ##"
30 A=0: PRINT" SZXHXPNC": INPUT"";A$: IF LEN(A$)<>8 THEN 20
40 FOR I=1 TO 8: X=VAL(MID$(A$,I,1))
50   IF X<>0 AND X<>1 THEN 20
60   A=A+A+X
70 NEXT: PRINT" = ";A: GOTO 30
```

Schreiben Sie ein Programm, das die dezimalen Zahlen des Flagregisters auf die HEX-Zahlen und das Bitmuster vorwärts und rückwärts umrechnet.

Sendung 3

Den Hauptteil aller Befehle machen die Ladebefehle aus. Gemäß den beiden Tabellen werden unterschieden: Tabelle auf Seite

- 8-Bit-Ladebefehle 1
- 16-Bit-Ladebefehle 2
- PUSH- und POP-Befehle, Ablage und Zurückholen vom Stack 2
- Sonstige Ladebefehle, z.B. Austauch EXX; EX DE,HL; LDIR 2

Eine andere Unterscheidung betrifft womit geladen wird:

- mit festen Werten: LD A,79D -> 3E 46 : LD BC,3145H -> 01 45 31
- zwischen Registern: LD A,B -> 78 : EX DE,HL -> EB
- zwischen Registern und Speichern LD A,(Adresse); LD HL,(Adresse):
  LD A,(3123H) -> 3A 23 13 : LD HL,(3465H) -> 2A 65 34

Bei den Tabellen S. 1 und 2 fällt auf, daß nicht alle Felder belegt sind. Bei sehr hoher Leistungsfähigkeit ist der U 880/ Z 80 leider unsymmetrisch im Befehlssatz. Dies erschwert etwas,ihn auswendig zu lernen.
Mit dem Programm REGIST kann der Inhalt aller Register angezeigt werden.

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## REGIST ##": PRINT
20 DOKE 772, 12288
30 FOR I=1 TO 7: READA$: PRINTA$;
40 FOR X=12288 TO 16388: READ A$
50   IF A$="EN" THEN X=16388: NEXT: GOTO 100
60   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
70   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
80   POKE X,A*16+B
90 NEXT
100 Y=USR(X):IF Y<0 THEN Y=65536+Y
110 Z=INT(Y/256): Y=Y-256*Z: PRINT Z;":";Y
120 NEXT
130 DATA "A:F=",F5,C5,F5,C1,79,CD,B1,D0,C1,F1,C9,EN
140 DATA "C:B=",F5,C5,79,CD,B1,D0,C1,F1,C9,EN
150 DATA "E:D=",F5,C5,D5,7B,42,CD,B1,D0,D1,C1,F1,C9,EN
160 DATA "L:H=",F5,C5,E5,7D,44,CD,B1,D0,E1,C1,F1,C9,EN
170 DATA "IX =",F5,C5,DD,E5,DD,E5,C1,79,CD,B1,D0,DD,E1,C1,F1,C9,EN
180 DATA "IY =",F5,C5,FD,E5,FD,E5,C1,79,CD,B1,D0,FD,E1,C1,F1,C9,EN
190 DATA "SP =",F5,C5,ED,73,00,00,3A,01,00,47,3A,00,00,CD,B1,D0,C1,F1,C9,EN
```

Am Ende des Programms stehen acht DATA-Zeilen, für je ein Register-paar bzw. ein 16-Bit-Register. Sie werden nacheinander ab 3000H abgelegt und dann aufgerufen. Zu ihen gehören die im folgenden Maschinenprogramme in Mnemoniks (Assembler) geschrieben

```
----AF----------BC---------DE------HL--------IX-----IY ------SP-----
PUSH  AF     PUSH  AF   PUSH  AF          PUSH  AF         PUSH  AF
PUSH  BC     PUSH  BC   PUSH  BC          PUSH  BC         PUSH  BC
PUSH  AF     LD    A,C  PUSH  DE    HL    PUSH  IX   IY    LD    (0),SP
POP   BC     CALL  AUS  LD    A,E   A,L   PUSH  IX   IY    LD    A,(1)
LD    A,C    POP   BC   LD    B,D   B,H   POP   BC         LD    B,A
CALL  AUS    POP   AF   CALL  AUS         LD    A,C        LD    A,(0)
POP   BC     RET        POP   DE    HL    CALL  AUS        CALL  AUS
POP   AF                POP   BC          POP   IX   IY    POP   BC
RET                     POP   AF          POP   BC         POP   AF
                        RET               POP   AF         RET
                                          RET
```

Beispielhaft sei die Wirkung für das Registerpaar DE besprochen:
Der Reihe nach werden die folgenden Registerpaare auf dem Stack abgelegt: AF, BC, DE. Dies ist notwendig, damit nach dem Programm alle Register wieder ihre Ausgangswerte durch Zurückholen mit POP erhalten können. Diese Methode des Rettens der verwendeten Register sollten Sie sich zur Pflicht machen. Denn durch sie gewinnt man Sicherheit für viele Anwendungen und für die Routinen u.a. des Betriebssystem. In unserem Falle für die exakte Weiterarbeit von BASIC nach der Rückkehr aus der Maschinenroutine. Denn natürlich nutzt BASIC auch die Register!
Nach den PUSH-Befehlen wird das Register E auf A und das Register D auf B geladen. A und B werden dann mit der Routine AUS an den Ausgangswert der USR-Funktion, also Y übergeben werden. Den Rest, nämlich die Anzeige, bewirkt wieder BASIC. Hier ist gemäß Zeile 100 ein eventueller negativer Wert (für größer 32768) noch in den positiven zu verwandeln (Y=Y+65536). Dann werden beide Anteile für D und E gemäß Zeile 120 getrennt zur Anzeige gebracht.
Die anderen Maschinenprogramme sollten Sie nach dieser Methode entsprechend selbst analysieren. Versuchen Sie weiter das Programm so zu ändern, daß es die Schattenregister anzeigt. Sie brauchen zusätzlich die Befehle: EXX, EX AF,AF'. Der erste schaltet zwischen Haupt- und Schattenregister für BC, DE und HL um, der zweite bzgl. A und F.
Bei der Registeranzeige wird vom Flagregister nur der Zahlenwert angezeigt. Es ist aber oft notwendig, über die einzelnen Bit Bescheid zu wissen. Hierzu dient das Programm FLAG. Die zusätzliche Aufarbeitung wurde hier in BASIC vorgenommen. Die Bedeutung der Flag enthält S.14.

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## FLAG ##": PRINT
20 DOKE 772, 12288
30 FOR X=12288 TO 16388: READ A$
40   IF A$="EN" THEN X=16388: GOTO 90
50   A=ASC(LEFT$(A$,1))-48: IF A>9 THEN A=A-7
60   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
70   POKE X,A*16+B
80 NEXT
90 Y=USR(X):IF Y<0 THEN Y=65536+Y
100 Z=INT(Y/256): PRINT "F =";Z: Q=128: PRINT
110 IF Z<Q THEN PRINT"S +": ELSE Z=Z-Q: PRINT"S -"
120 Q=Q/2: IF Z<Q THEN PRINT" NZ": ELSE Z=Z-Q: PRINT"  Z"
130 Q=Q/2: IF Z<Q THEN PRINT"N 5": ELSE Z=Z-Q: PRINT"J 5"
140 Q=Q/2: IF Z<Q THEN PRINT"N H": ELSE Z=Z-Q: PRINT"J H"
150 Q=Q/2: IF Z<Q THEN PRINT"N 3": ELSE Z=Z-Q: PRINT"J 3"
160 Q=Q/2: IF Z<Q THEN PRINT" PO": ELSE Z=Z-Q: PRINT" PE"
```

```
170 Q=Q/2: IF Z<Q THEN PRINT"N N": ELSE Z=Z-Q: PRINT"J N"
180 Q=Q/2: IF Z<Q THEN PRINT" NC": ELSE PRINT"  C"
190 : PRINT: PRINT
200 DATA F5,C5,F5,C1,78,CD,B1,D0,C1,F1,C9,EN
210 !------------- Anzeige der Flag-Werte ----------------
```

Quellcode:		Flaganzeige:
PUSH	AF	F = 58
PUSH	BC	S +
PUSH	AF	NZ
POP	BC	J 5
LD	A,B	J H
CALL	AUS	J 3
POP	BC	PO
POP	AF	J N
RET		NC

Abschließend für diese Sendung sei noch der mächtiger Befehl **LDIR** verwendet: er lautet etwa: lade, decrementiere und laufe (run). Hierzu werden gewählt:

HL	verweist auf eine Speicheradresse von der ein Wert nach
DE	Adresse übertragen wird
BC	weist die Länge eines Speicherbereiches aus

Nach dem Übertragen des Inhaltes von Adresse HL auf die Adresse von DE werden HL, DE und BC um je eins erniedrigt (decrementiert), Danach wird geprüft, ob BC Null ist. Wenn das der Fall ist, ist der Befehl beendet. Andernfalls wird der nächste Wert von (HL) nach (DE) übertragen und wieder decrementiert usw. Dieses wird im Programm MOVE genutzt.

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## MOVE ##": PRINT
20 FOR X=12288 TO 16388: READ A$
30   IF A$="EN" THEN X=16388: NEXT: GOTO 80
40   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
50   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
60   POKE X,A*16+B
70 NEXT
80 DOKE 772, 12288: CALL*3006
90 DATA 43,44,71,83,84,85
100 DATA E5,D5,C5,21,00,30,11,20,30,01,06,00,ED,B0,C1,D1,E1,C9,EN
```

Quelle für Zeile 100:

```
PUSH   HL              LDIR
PUSH   DE              POP    BC
PUSH   BC              POP    DE
LD     HL,3000H        POP    HL
LD     DE,3020H        RET
LD     BC,6
```

Es werden zunächst die Werte von Zeile 90 ab 3000H abgelegt. Mit dem Maschinenprogramm ab 3006H werden diese Werte dann zusätzlich hinter 3020H kopiert. Dies können Sie sich nur im CAOS-Betriebssystem mit DISPLAY oder im Zusatzmonitor mit D 3000_302F nach Ablauf des BASIC-Programmes ansehen.

Sendung 4

Schwerpunktmäßig gilt diese Sendung den arithmertischen Befehlen. Es werden unterschieden, geordnet nach 8-Bit und 16-Bit

Beschreibung	Beispiel 8-Bit		Beispiel 16-Bit	
Addition ohne C-Flag	ADD A,4	C6 04	ADD HL,BC	09
Addition mit C-Flag	ADC A,(IX+3)	DD 8E 03	ADC HL,SP	ED 7A
Subtraktion ohne C-Flag	SUB A,(HL)	96	nicht vorhanden	
Subtraktion mit C-Flag	SBC A,L	9D	SBC HL,BC	ED 42
Logische Operationen	OR B	B0	nicht vorhanden	
um 1 Erhöhen, Erniedrigen	INC E	1C	DEC HL	2B

Das indirekte Register (HL), Inhalt von HL wird auch oft mit M bezeichnet.
Das relativ einfache Programm DREIFA gibt von einer eingegebenen ganzen Zahl im Bereich von 0 bis ca. 20000 den dreifachen Wert zurück:

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## DREIFA ##": PRINT
20 FOR X=12288 TO 16388: READ A$
30   IF A$="EN" THEN X=16388: NEXT: GOTO 80
40   A=ASC(LEFT$(A$,1))-48: IF A>9 THEN A=A-7
50   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
60   POKE X,A*16+B
70 NEXT
80 DOKE 772, 12288
90 INPUT"Zahl =";X: Y=USR(X): PRINT Y: GOTO90
100 ! Uebergibt Zahl an USR und die 3-fache zurueck
110 ! EIN=C96F:X->DE: AUS=D0B1: AB->Y
120 ! DOKE 772, Adresse legt fest, wo Maschinenprogramm liegt
130 DATA F5, C5, D5, E5, CD, 6F, C9, D5, E1, 19, 19, 7C, 45
140 DATA CD, B1, D0, E1, D1, C1, F1, C9, EN
```

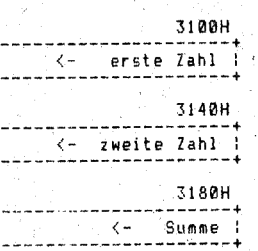
Das Maschinenprogramm hierzu funktioniert wie folgt: Nach dem Wegpushen der Register AF, BC, DE und HL wird mit der Routine EIN der Wert X von USR(X) an das Registerpaar DE übergeben. Mit PUSH DE und POP HL wird er zusätzlich auf HL übertragen. Dann wird zweimal mit ADD HL,DE der Wert dazuaddiert. Danach sind die Inhalte mit LD A,H und LD B,L auf die Register A und B zu übertragen. Von hier können sie mit der Routine AUS auf Y von Y=USR(X) an BASIC zurückgegeben und PRINT Y angezeigt werden.
In diesem Beispiel wurde einmal ein Ausdruck gewählt, wie ihn etwa ein Assembler (Tabelle S. 3) in etwa erzeugt:

```
F5       PUSH   AF              7C       LD     A,H
C5       PUSH   BC              45       LD     B,L
D5       PUSH   DE              CDB1D0   CALL   AUS
E5       PUSH   HL              E1       POP    HL
CD6FC9   CALL   EIN             D1       POP    DE
D5       PUSH   DE              C1       POP    BC
E1       POP    HL              F1       POP    AF
19       ADD    HL,DE           C9       RET
19       ADD    HL,DE
```

Eine wesentlich leistungsfähigere Variante ist das Programm ADIT:

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## ADII ##": PRINT
20 FOR X=12288 TO 16388: READ A$
30   IF A$="EN" THEN X=16388: NEXT: GOTO 80
40   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
50   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
60   POKE X,A*16+B
70 NEXT
80 PRINT: CALL*3000: K=12544:!3100H; loeschen
90 FOR I=1 TO 2
100   PRINT"Zahl";I;: INPUT"=";A$: L=LEN(A$): M=L/2
110   IF M>INT(M) THEN A$="0"+A$: L=L+1
120     FOR J=L TO 1 STEP -2
130       POKE K-(L-J)/2, VAL(MID$(A$,J,1))+16*VAL(MID$(A$,J-1,1))
140     NEXT: K=K+64
150 NEXT: CALL*3016: F=0:! addieren; F fuer 0-Unterdrueckung
160 FOR J=12609 TO K:! =3141H
170   Y=PEEK(J): IF Y THEN F=1
180   IF F=0 THEN NEXT: GOTO 80
190   IF Y<0 THEN Y=65536+Y
200   Z=INT(Y/16): Y=Y-16*Z: PRINT Z*10+Y;
210 NEXT: GOTO 80
220 DATA F5,C5,D5,E5,21,80,30,01,00,01,AF,77,E5,D1,13,ED,B0,E1,D1,C1,F1,C9
230 DATA F5,C5,E5,D5,FD,E5,06,40,21,00,31,11,40,31,FD,21,80,31
240 DATA AF,1A,8E,27,FD,77,00,2B,1B,FD,2B,10,F4,FD,E1,D1,E1,C1,F1,C9,EN
250 !---addiert vielstellige Zahlen---
```

Es können hierbei sehr lange ganze, positive Zahlen (bis über 70 Stellen) eingegeben werden. Sie werden ab 3100H = 12544D bzw. 3140H = 12708D rückwärts, also zu kleineren Adressen hin abgelegt. Hierzu gilt das Bild



Die Ablage erfolgt gemäß Zeile 130 in BCD (binär codiertes dezimal). Dies bedeudet, daß im rechten und linken Halbbyte (nibble) je eine Ziffer zwischen 0 und 9 steht. Mit der Maschinenroutine werden sie addiert und ab 3180H rückwärts abgelegt. Die Addition von BCD-Zahlen verlangt den DAA-Befehl (dezimal justieren der Addition) nach jeder Addition.
Im Maschinenprogramm steht zunächst in Zeile 220 der Code zum Löschen des gesamten Speichers.

```
PUSH   AF
PUSH   BC
PUSH   DE
PUSH   HL
LD     HL,3080H
LD     BC,100H     ; Speicherbereich
XOR    A           ; A=0
LD     (HL),A      ; erste Zelle mit 0 belegen
PUSH   HL
POP    DE
INC    DE          ; DE=HL+1
LDIR               ; alle Speicherwerte werden 0 gesetzt
POP    HL
POP    DE
POP    BC
POP    AF
RET
```

Ab Zeile 230 folgt dann die eigentliche Additionsroutine. Sie liegt ab der Adresse 3016H und funktioniert so wie eine schriftliche Addition, wobei der eventuelle Obertrag im C-Flag zwischendurch gespeichert wird. Deshalb wird zu Beginn mit XOR A das C-Flag auch zurückgesetzt. Eine zusätzliche Erklärung bedarf noch der Befehl DJNZ MARKE.
Hierbei wird das B-Register um eins vermindert. Danach wird geprüft, ob es Null ist. Wenn dies der Fall ist, wird zur nächsten Zeile übergegangen. Andernfalls wird zur MARKE zurückgesprungen. Dies ist ein relativer Sprung, der gemäß der Tabelle 12 bestimmt werden kann.
Im BASIC-Programm werden die einzelnen Byte über PEEK (Zeile 170) zurückgeholt. Für die Ausgabe, welche im Adreßraum jetzt von links nach rechts erfolgt, müssen die Anfangsnullen unterdrückt werden. Dies wird mitttels der Variablen F realisiert.

```
      PUSH  AF                          MARKE  LD    A,(DE)
      PUSH  BC                                 ADC   A,(HL)
      PUSH  HL                                 DAA
      PUSH  DE                                 LD    (IY),A
      PUSH  IY                                 DEC   HL
      LD    B,40H                              DEC   DE
      LD    HL,3100H                           DEC   IY
      LD    DE,3140H                           DJNZ  MARKE
      LD    IY,3180H                           POP   IY
      XOR   A       ;C-FLAG = 0                POP   DE
                                               POP   HL
                                               POP   BC
                                               POP   AF
                                               RET
```

Abschließend für diese Sendung sei noch ein Beispiel für die Bit-Befehle vorgestellt. Das Maschinenprogramm legt ab HL=3020H alle HEX-Werte in der Reihenfolge von B, also 00, FF, FE, FD, ..., 02, 01 ab. Dies erfolgt wieder mittels DJNZ Marke. Dann wird mit POP DE der Anfangswert der Ablage zurückgeholt. Mit LD A,(DE) erfolgt dies der Reihe nach. Mit AND 7H werden von diesen Werten nur die letzten drei Bit beibehalten. Man sagt AND 7H wirkt hier als Maske. Mit OR 30H werden Bit 4 und 5 gesetzt. So können nur die Werte 30H bis

39H entstehen und dies sind gerade die ASCII-Code für die Zahlen 0 bis 9. Das Ergebnis ist nur im Maschinencode zu besichtigen.
Im Maschinencode ist eine leichte Modifikation angedeutet. Dabei werden Bit 7 zurückgesetzt und Bit 6 gesetzt. Führen Sie bitte diese Änderung durch und prüfen Sie was dann geschieht!

```
       PUSH  AF
       PUSH  BC
       PUSH  DE
       PUSH  HL
       LD    B,0
       LD    HL,3020H
       PUSH  HL          ;rettenfür später
MARKE  LD    (HL),B
       INC   HL
       DJNZ  MARKE
       POP   DE          ;Adresse zurück
MARKE2 LD    A,(DE)      ;für nur ASCII
       AND   7H          ;(RES  7,A)
       OR    30H         ;(SET  6,A)
       LD    (HL),A
       INC   HL
       INC   DE
       DJNZ  MARKE2
       POP   HL
       POP   DE
       POP   BC
       POP   AF
       RET
```

Das zugehörige BASIC-Programm lautet:

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## ZAHL ##": PRINT
20 FOR X=12288 TO 16388: READ A$
30 IF A$="EN" THEN X=16388: NEXT: CALL*3000: END
40   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
50   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
60   POKE X,A*16+B
70 NEXT
80 DATA F5,C5,D5,E5,06,00,21,20,30,E5,70,23,10,FC,D1,1A,E6,07,F6,30
90 DATA 77,23,13,10,F6,E1,D1,C1,F1,C9,EN
100 'erzeugt aus allen HEX-Werten Zahlen
```

Sendung 5

Diesmal werden direkte Zugriffe auf den Bildschirm erfolgen. Deshalb ist zwischen den KC 85/2 bis 4 und den KC 85/1 sowie KC 87 zu unterscheiden. Beide Geräteklassen haben hierfür Vor- und Nachteile.

KC 85/2 bis 4
- Direkt kann nur auf den Pixel-RAM von 8000H bis A800H zugegriffen werden
- Es ist der Bildschirm-RAM für den Zugriff einzuschalten

KC 85/1 und KC 87
- Der Bildschirm enthält nur HEX-Code und liegt von EC00H bis EFC0H

Mit dem ersten Programm werden zunächst alle HEX-Werte übergeben. Zur Unterscheidung beider Gerätereihen wird die Speicherstelle -5 abgefragt. Sie kennzeichnet erfahrungsgemäß beide Rechnertypen. Die beiden etwas unterschiedlichen Maschinenprogramme sind in den DATA-Zeilen 90 und 100 abgelegt. Für die KC 85/1 und KC 87 wird das Programm mit RESTORE 100 ausgewählt.

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## BILD ##": PRINT
20 IF PEEK(-5)>129 THEN RESTORE 100
30 FOR X=12288 TO 16388: READ A$
40 IF A$="EN" THEN X=16388: NEXT: CALL*3000: END
50   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
60   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
70   POKE X,A*16+B
80 NEXT
90 DATA C5,E5,DB,88,CB,D7,D3,88,06,00,21,00,80,70,23,10,FC,E1,C1,C9,EN
100 DATA C5,E5,06,00,21,00,EC,70,23,10,FC,E1,C1,C9,EN
110 !Erzeugt einfaches Muster
```

Der Maschinencode dafür leitet sich aus der folgenden Quellen ab. Für den KC 85/2 bis 4 ist hierbei auch einmal der IN- und OUT-Befehl genutzt. Die PIO ist bitweise organisiert und mit der Adresse 88H kann von Bit 2 der Bildschirm an- und abgschaltet werden. Mit IN A,(88H) wird der aktuelle Wert geholt, und SET 2,A schaltet den Bildschirm bei der Rückgabe OUT (88H),A an.

```
        PUSH  BC                    ;Für KC 87
        PUSH  HL
        IN    A,(88H)               ;entfaellt für KC 87
        SET   2,A                   ;IMR AN
        OUT   (88H),A               ;entfaellt
        LD    B,0
        LD    HL,8000H    HL,EC00H
MARKE   LD    (HL),B
        INC   HL
        DJNZ  MARKE
        POP   HL
        POP   BC
        RET
```

Wesentlich leistungsfähiger ist das folgende Programm. Es setzt einen irgendwie beschrieben Bildschirm voraus. Deshalb fehlt hier der Löschbefehl in Zeile 10. Beim KC 85/2 bis 4 werden noch zusätzlich Kreise geschrieben.

```
10 CLEAR 100, 12280: PRINT TAB(15);"## BILDVAR ##": PRINT
20 IF PEEK(-5)>129 THEN RESTORE 120
30 IF PEEK(-5)<130 THEN: FOR X=1 TO 400 STEP 10: CIRCLE 7+X,8+X,X,7: NEXT
40 FOR X=12288 TO 16388: READ A$
50 IF A$="EN" THEN X=16388: NEXT: CALL*3000
60   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
70   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
80   POKE X,A*16+B
90 NEXT
100 DATA F5,C5,E5,DB,88,CB,D7,D3,88,01,00,28,21,01,80,7E,23,AE,23,AE
110 DATA 2B,77,0B,78,B1,20,F4,18,EC,EN
120 DATA F5,C5,E5,01,C0,03,21,00,EC,7E,23,AE,23,AE
130 DATA 2B,77,0B,78,B1,20,F4,18,EC,EN
140 !Erzeugt bewegtes Bild
```

Das Maschinenprogramm tastet den gesamten Bildschirm ab. Für jeden Speicherplatz werden zusätzlich zwei Nachbarwerte geholt und über XOR (HL) verknüpft. Das Ergebnis wird dann zurückgespeichert. Dieser Vorgang wird ständig wiederholt. Der Abbruch des Programms ist nur mit der RESET-Taste möglich.
Bei den KC 85/1 und KC 87 entsteht entsprechend dem Maschinenprogramm ein sehr schneller Wechsel der ASCII- und Pseudografik-Zeichen.
Bei den KC 85/2 bis 4 wird das Bild ständig variert und kehrt zwischendurch fast originalgetreu wieder. Es lohnt sich, das Bild länger zu beobachten. Im Vergleich von KC 85/2 und 3 gegenüber dem KC 85/4 wird der veränderte Bildschirmaufbau deutlich. Bemühen Sie sich bitte, dies Maschinenprogramm für beide Rechner zu verstehen. Bezüglich der beiden Rücksprünge seien noch kleine Hinweise gegeben. Die Größe des Bildschirmes wird in beiden Programmvarianten an BC übergeben (ANFANG). Der Wert wird mit DEC BC reduziert, und jedesmal wird danach abgefragt, ob er bereits Null ist. Dies erfolgt, indem zunächst B nach A umgespeichert (LD A,B) und dann mit OR C kombiniert wird. Nur wenn danach A null ist (Z-Flag) ist auch BC null und es erfolgt der Rücksprung zu ANFANG. Andernfalls erfolgt der Rücksprung zu MARKE. Die Berechnug der Rücksprungzahlen erfolgt in der nächsten Sendung.

```
        PUSH  AF                        ;Fuer KC 87
        PUSH  BC
        PUSH  HL
        IN    A,(88)                    ;streichen bei KC 87
        SET   2,A                       ;IMR AN
        OUT   (88),A                    ;streichen
ANFANG  LD    BC,1800H    BC,3C0H       ;960D
        LD    HL,8000H    HL,EC00H      ;-5120D
MARKE   LD    A,(HL)
        INC   HL
        XOR   (HL)
        INC   HL
        XOR   (HL)
        DEC   HL
        LD    (HL),A
        DEC   BC
        LD    A,B
        OR    C
        JR    NZ, MARKE
        JR    ANFANG
```

Sendung 6

Das hier zu besprechende Programm SUTEXT sucht ab einer vorgebbaren Speicheradresse nach den ASCII-Zeichen für Buchstaben.

```
        PUSH    AF
        PUSH    BC
        PUSH    HL
        LD      HL,(0)
MARKE   INC     HL
        LD      A,(HL)                  wenn alle ASCII
        RES     5,A         ;nur gross  BIT     7,A         ; groesser 127
        CP      5BH         ;'Z'+1      JR      NZ,MARKE
        JR      NC,MARKE                CP      20H         ;Steuerzeichen
        CP      41H         ;'A'        JR      C,Marke
        JR      C,MARKE                 NOP
        LD      B,(HL)
        XOR     A           ; gleich Null
        LD      (0),HL
        CALL    AUS
        POP     HL
        POP     BC
        POP     AF
        RET
```

Mit LD HL,(0) wird zunächst aus dem Speicherzellen 0 und 1 die Startadresse geholt. Mit INC HL und LD A,(HL) wird von dort das zu untersuchende Zeichen geladen. Um die kleinen und großem Buchstaben zu finden werden zunächst mit RES 5,A alle Kleinbuchstaben zu Großbuchstaben gewandelt. Dann erfolgt ein Vergleich mit dem ASCII-Code von "Z" (um eins erhöht). Alle Zeichen die "größer" als "Z" sind werden mit JR NC,MARKE übersprungen. Für die dann noch bleibenden Zeichen erfolgt ein Vergleich mit "A". Alle die kleiner sind werden mit JR C,Marke übersprungen. Die gültigen Zeichen werden dann direkt auf B mit LD B,(HL) übertragen. Wegen der Obergabe an BASIC wird außerdem A Null gesetzt (XOR A) und schließlich wird die Adresse noch in 0 und 1 abgelegt. MIT der Routine AUS geht der Wert zurück ins BASIC.

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## SUTEXT ##": PRINT
20 FOR X=12288 TO 16388: READ A$
30   IF A$="EN" THEN X=16388: NEXT: GOTO 80
40   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
50   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
60   POKE X,A*16+B
70 NEXT
80 INPUT"Start dezimal =";A: DOKE 0, A
90 C=65536: DOKE 772, 12288:! USR
100 Y=USR(X): A$="": D=C: A=DEEK(0):IF A<0 THEN A=C+A:! Zeichen, Adresse
110 FOR I= 1 TO 4: D=D/16
```

```
120 B=INT(A/D): A=A-B*D: B=B+48: IF B>57 THEN B=B+7
130 A$=A$+CHR$(B)
140 NEXT: PRINT A$;" =";DEEK(0);" : ";CHR$(Y)
150 Z=Z+1: IF Z<20 THEN 100: ELSE Z=0: INPUT"";A: GOTO 100
160 DATA F5,D5,E5,2A,00,00,23,7E,CB,AF,FE,5B,30,F8
170 DATA FE,41,38,F4,46,AF,22,00,00,CD,B1,D0,E1,D1,F1,C9,EN
180 '--- sucht Buchstaben ---------
```

Die Adresse wird in Zeile 100 mit A=DEEK(0)geholt und sowohl nach Wandlung (Zeilen 110 bis 140) in HEX-Code als Dezimal mit dem zugehörigen Zeichen angezeigt. In Zeile 150 werden so 20 Zeichen angezeigt. Dann muß mit ENTER oder einer Zahl erst das INPUT zur weiteren Anzeige überwunden werden.
Die Startadresse wird in Zeile 80 dezimal eingegeben. Für die Analyse des BASIC-Programms ist 1023D = 3FFH einzugeben. Mit der Eingabe -16385D=C000H kann der BASIC-Interpreter untersucht werden. Hiebei können z.B. die Adressen für BASIC-Start und -Restart, aber auch für Token und Textketten gefunden werden, was sich dann im Monitor weiter verfolgen läßt. Mit -8193D=E000H kann das Monitorprogramm durchsucht werden.
Im Maschinencode ist für Sie eine Modifikation angezeigt, mit der Sie alle ASCII-Zeichen erhalten. Versuchen Sie so das Programm zu verändern. Überlegen Sie einmal weiter wie Sie alle Token finden können. Bei ihnen ist Bit 7 gesetzt, Sie können auch abfragen, ob ein Zeichen größer als 127D = 7FH ist.
An diesem Programm ist gut das Fehlerverhalten von falschen Programmen zu testen. Sie brauchen nur die Bedingungen für die Rücksprünge, also JR NC und JR C (30 und 38) zu vertauschen und es geschieht scheinbar nichts mehr. Das Programm fängt sich in einer Schleife, ohne daß nach außen etwas sichtbar ist. Versuchen Sie dies zu erklären. Das Programm können Sie dann nur mit RESET unterbrechen. Dabei bleibt das BASIC-Programm erhalten. Sie können die Hex-Code in den DATA-Zeilen ändern. Hier zeigt sich, daß das hier benutzte Prinzip über BASIC relativ stabil und gut editierbar ist.
Abschließend noch Bemerkungen zur Berechnung der Relativsprünge. Es gelten die die folgende Zählregeln

vorwärts verweisend	rückwärts verweisend
JR +10 (0A)	# Hierher springen XX
XX	XX XX XX
XX XX XX	XX XX
XX XX	XX XX XX
XX	XX
XX	XX
XX XX	JR -13 (F3)
# Hierher springen	Hier werden also JR und -13 mitgezählt

Die Umrechnungen von diesen Zählwerten in den HEX-Code leistet die Tabelle auf S. 12. Es ist das Ergebnis aber auch in beiden Fällen von "Hand" nachprüfbar. Bei vorwärts muß man nur echt in HEX zählen, also 1, 2, 3, ..., 9, A, B, C, ..., F, 10, 11,..., 1F, 20 usw. Die Rücksprünge beginnen mit dem Ergebnis oben F3 und müssen dann bei -13 mit FF enden.

Dies können Sie bei dem folgenden Programm MOVUN erproben. Es geht von einer Quell-Adresse aus. Von ihr an soll ein Code einer bestimmten Länge von einer Ziel-Adresse an hin verlagert werden. Damit dies immer fehlerfrei funktioniert muß das Programm zunächst prüfen, ob die Ziel-Adresse größer als die der Quell-Adresse ist. Je nachdem muß die Verschiebung von unten mit LDIR oder von oben mit LDDR erfolgen. Für die Übergabe der Werte an den Maschinencode werden die Speicherzellen 0 bis 5 gemäß dem folgenden Schema benutzt:

```
   0      1       2       3       4       5
+------------+-------------+------------+
| Quelle     |  Ziel       |  Länge     +
+------------+-------------+------------+
```

Das Maschinenprogramm lautet dann:

```
        PUSH    BC                      ;Ziel <= Quelle
        PUSH    DE                              LDIR
        PUSH    HL                              JR      ENDE
        LD      HL,(0)                  ;Ziel > Quelle
        LD      DE,(2)                  MARKE   ADD     HL,BC
        LD      BC,(4)                          EX      DE,HL
        PUSH    HL                              ADD     HL,BC
        OR      A       ; RESET C-Flag          EX      DE,HL
        SBC     HL,DE                           LDDR
        POP     HL                      ENDE    POP     HL
        JR      C,MARKE                         POP     DE
                                                POP     BC
                                                RET
```

Für das zugehörige BASIC-Programm gilt:

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## MOVUN ##": PRINT
20 FOR X=12288 TO 16388: READ A$
30   IF A$="EN" THEN X=16388: NEXT: GOTO 80
40   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
50   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
60   POKE X,A*16+B
70 NEXT
80 FOR I=0 TO 2: READ A$: PRINT A$;: INPUT"=";A$: B=0
90   FOR J=1 TO LEN(A$)
100    A=ASC(MID$(A$,J,1))-48: IF A>9 THEN A=A-7
110    B=B*16+A
120  NEXT: IF B>32768 THEN B=B-65536
130 DOKE I+I,B
140 NEXT: CALL*3000
150 DATA C5,D5,E5,2A,00,00,ED,5B,02,00,ED,4B,04,00,B7,E5,ED,52,E1,38,04
160 DATA ED,B0,18,06,09,EB,09,EB,ED,B8,E1,D1,C1,C9,EN
170 DATA Quelle, Ziel, Laenge
180 !Verschiebt beliebig Codes
```

Sendung 7

Bisher wurden alle Maschinenprogramme per "Hand" mit den Tabellen erzeugt. Dies ist effektiv nur bis etwa 100 Byte durchführbar. Bei größeren Programmen ist immer ein Assembler angebracht. Hier wird zunächst ein Quelltext erzeugt, der in etwa den bisherigen Maschinenprogrammen mit Mnemoniks entspricht. Er erhält in der Regel einen Namen, wie z.B. SUCHE.ASM, wobei die 3 Zeichen hinter dem Punkt die nähere Kennzeichnung (Extension) sind. Je nach dem verwendeten Assembler können auch andere Buchstaben stehen. Unter diesem Namen wird der Quelltext nach dem Schreiben zunächst gespeichert. Dann wird der Assembler mit diesem Programm zusammen aufgerufen und meist erst auf einfache Schreibfehler (Syntax) getestet. Wenn hierbei keine Fehler mehr gefunden werden kann der Quelltext assembliert werden. Dabei entsteht ein Listing der folgenden Art (gemacht mit EDAS):

```
6000            ;--------------------
6000            ;M-Code fuer Suche  !
6000            ; H.Voelz  Dez.1988 !
6000            ;--------------------
6000            ;
6000                    ORG     3000H
3000            ;
3000 F5                 PUSH    AF
3001 E5                 PUSH    HL
3002 2A0000             LD      HL,(0)
3005 23         MARKE   INC     HL
3006 3A0200     MARKE2  LD      A,(2)
3009 BE                 CP      (HL)
300A 20F9               JR      NZ,MARKE
300C            ;Beides gleich
300C 23                 INC     HL
300D 3A0300             LD      A,(3)
3010 BE                 CP      (HL)
3011 20F0               JR      NZ,MARKE2
3013            ;Kette gefunden
3013 2B                 DEC     HL
3014 220000             LD      (0),HL
3017 E1                 POP     HL
3018 F1                 POP     AF
3019 C9                 RET
```

Hierin bedeuten

```
|     |       |       |       |
|     |       |       |       Mnemonik Teil 2
|     |       |       Mnemonik Teil 1
|     |       Markenfeld, entspricht etwa anzuspringender Zeile in BASIC
|     Objektcode
Adresse für die Ablage des Objektcodes.
```

Der Start wird durch die Ursprungsanweisung ORG (origin) festgelegt. Auffällig ist weiter der Kopf mit Hinweisen auf das Programm, das Datum und dem Programmierer. Oft ist mehr notwendig und sinnvoll. Alle Kommentare, also

auch der Kopf, werden bei den meisten Assemblern hinter Semikolon geschrieben. Ähnlich wie in BASIC hinter ! oder REM.
Der eigentliche Quelltext besteht nur aus dem Markenfeld und den beiden Mnemonikteile, sowie den Kommentaren. Den Adreßteil und Objektcode fügt der Assembler für das obige Listing (Drucktext) hinzu. Auch er wird oft gespeichert, allerdings mit einer anderen Extension z.B. PRT. Er wird in der Regel nur zu Dokumentation verwendet. Mit einem weiteren Programm (link) kann der Objektcode automatisch an die richtigen Stellen abgelegt werden. Seine Speicherung erfolgt ebenfalls unter einem Namen, z.B. mit der Extesion REL. Es entfällt also das Eingeben in BASIC-DATA-Zeilen. Mit weiteren Hilfmitteln kann dann der Objektcode aufgerufen werden. Genau diese Vielfalt der Operationen erfordert eine gewisse Erfahrung. Sie wurde hier durch die BASIC-Methode umgangen. Aber gewiß dürfte es Ihnen jetzt leichter fallen, in der genannten Art zu operieren. Das Literaturverzeichniß wird Ihnen weiter helfen.
Das Programm ermöglicht die Suche nach zwei aufeinanderfolgenden HEX-Code im Sinne einer Adresse. Sie wird z.B. als C9DF eingegeben, im Speicher steht dann DF C9. Die Adressen, bei denen dies der Fall ist, werden angezeigt. Dabei kann der Speicher mehrfach durchlaufen werden. Dann wird auch immer die Adresse 2 vorkommen, weil hier der Suchcode ja abgelegt wurde.

```
  0        1          2         3
+------------------+---------------+
| aktuelle Adresse | Suchcode      |
+------------------+---------------+
```

Das zugehörige BASIC-Programm lautet:

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## SUCHE ##": PRINT
20 FOR X=12288 TO 16388: READ A$
30   IF A$="EN" THEN X=16388: NEXT: GOTO 80
40   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
50   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
60   POKE X,A*16+B
70 NEXT
80 FOR I=0 TO 1: READ A$: PRINT A$;: INPUT"=";A$: B=0
90   FOR J=1 TO LEN(A$)
100    A=ASC(MID$(A$,J,1))-48: IF A>9 THEN A=A-7
110    B=B*16+A
120  NEXT: IF B>32768 THEN B=B-65536
130 DOKE I+I,B
140 NEXT
150 CALL*3000: PRINT"CRC =";PEEK(4): RESTORE 180: GOTO 80
160 DATA F5,C5,D5,E5,ED,5B,00,00,2A,02,00,AF,32,04,00,ED,52,EB,3A,04,00
170 DATA 47,7E,90,47,A8,2F,90,32,04,00,23,1B,7A,B3,20,ED,E1,D1,C1,F1,C9,EN
180 DATA Start,"Ende "
190 !Sucht Codes, Reihenfolge low, high
```

Für die Kontrolle auf Änderungen von Programmen ist ein CRC (Cyclischer Redundanz-Code) nützlich. Er ist viel leistungsfähiger als eine Prüfsumme. Er wird sehr oft bei der Speicherung und Rückgabe von Programmen und Daten verwendet. Dazu wird sein Wert aus den Daten bei der Speicherung zusätzlich erzeugt und abgelegt. Bei der Wiedergabe geschieht das Gleiche und zum Schluß werden gespeichertes und neu berechnetes CRC-Wort vergliechen. Stimmen sie beide überein, so liegt mit sehr großer Wahrscheinlichkeit kein Fehler vor. Bei Abweichungen ist dagegegn ein Fehler gewiß. In der letzten Zeit hat diese

Methode auch bezüglich Viren, die ja das Programm und die Daten ändern, an Bedeutung gewonnen. Hierzu geht man wie folgt vor: Es wird bei der Programmerstellung das CRC-Wort gebildet und in gewissen Abständen wird es wieder geprüft. Stimmt es, so ist sehr wahrscheinlich das Programm unverändert. Oft werden nach diesem Prinzip auch die Inhalte von EPROM überprüft. Das folgende Maschinenprogramm erzeugt Ihnen auf einfache Weise ein 8-Bit-CRC. Das Grundprinzip ist im Adreßbereich von 3012H bis 301EH abgelegt. Leider kann diese Routine hier nicht erklärt werden. Es genügt bereits gewissen Ansprüchen.

```
6000            ;--------------------
6000            ;M-Code fuer CRC     !
6000            ; H.Voelz  Dez.1988 !
6000            ;--------------------
6000            ;
6000                    ORG     3000H
3000            ;
3000 F5                 PUSH    AF
3001 C5                 PUSH    BC
3002 D5                 PUSH    DE
3003 E5                 PUSH    HL
3004 ED5B0000           LD      DE,(0)
3008 2A0200             LD      HL,(2)
300B AF                 XOR     A
300C 320400             LD      (4),A
300F ED52               SBC     HL,DE
3011 EB                 EX      DE,HL
3012 3A0400     MARKE   LD      A,(4)
3015 47                 LD      B,A
3016 7E                 LD      A,(HL)
3017 90                 SUB     B
3018 47                 LD      B,A
3019 A8                 XOR     B
301A 2F                 CPL
301B 90                 SUB     B
301C 320400             LD      (4),A
301F 23                 INC     HL
3020 1B                 DEC     DE
3021 7A                 LD      A,D
3022 B3                 OR      E
3023 20ED               JR      NZ,MARKE
3025 E1                 POP     HL
3026 D1                 POP     DE
3027 C1                 POP     BC
3028 F1                 POP     AF
3029 C9                 RET
```

Das zugehörige BASIC-Programm lautet:

```
10 CLEAR 100, 12280: CLS: PRINT TAB(15);"## SUCHE ##": PRINT
20 FOR X=12288 TO 16388:READ A$
30   IF A$="EN" THEN X=16388: NEXT: GOTO 80
40   A=ASC(LEFT$(A$,1))-48:  IF A>9 THEN A=A-7
50   B=ASC(RIGHT$(A$,1))-48: IF B>9 THEN B=B-7
60   POKE X,A*16+B
70 NEXT
80 FOR I=0 TO 1: READ A$: PRINT A$;: INPUT"=";A$: B=0
90   FOR J=1 TO LEN(A$)
100    A=ASC(MID$(A$,J,1))-48: IF A>9 THEN A=A-7
110    B=B*16+A
120  NEXT: IF B>32768 THEN B=B-65536
130 DOKE I+I,B
140 NEXT
150 CALL*3000: PRINT DEEK(0): Z=Z+1: IF Z=20 THEN Z=0: INPUT"";A
160 GOTO 150
170 DATA F5,E5,2A,00,00,23,3A,02,00,BE,20,F9,23,3A,03,00,BE,20,F3
180 DATA 2B,22,00,00,E1,F1,C9,EN
190 DATA Start,Code
200 !Sucht Codes, Reihenfolge low, high
```

Die Anfangs- und Endadresse werden wieder in HEX eingegeben. Das CRC-Byte wird dezimal angezeigt.

Literatur
=========

[1] Kieser, H. u. Meder, M.: Mikroprozessortechnik. Verlag Technik, Berlin 1985
[2] Warme, G. u.a.: Mikrorechneranwendung. Institut für Nachrichtentechnik. Berlin 1982
[3] Claßen, L.: Programmierung des Mikroprozessorsystems U 880 - K 1520 Verlag Technik. Berlin 1985
[4] Völz, H.: Elektronik. Akademie-Verlag. Berlin 1986
[5] Völz, H.: Tabellen zum Mikrorechnersystem U 880 - Z 80. Akademie-Verlag Berlin 1986
[6] Barthold,H. u. Bäurich,H.: Mikroprozessoren, Mikroelektronische Schaltkreise und ihre Anwendung. Militärverlag. Berlin 1985
[7] N.N.: Beschreibung zum M027 Development für den KC 85. VEB Mikroelektronik Mühlhausen
[8] N.N.: Interpretativer Dialogassembler zum Z 9001. VEB Robotron-Meßelektronik Dresden

Programme

1 DATHEX , DATASI
2 DEZHEX , BITFLAG
3 REGIST , FLAG , MOVE
4 DREIFA , ADIT , ZAHL
5 BILD , BILDVAR
6 SUTEXT , MOVUN
7 SUCHE , CRC

Unsere Sendefrequenzen auf UKW :

	Radio DDR II	DT 64 (MHz)
Frankfurt/O.	87,6	101,5
Putbus	88,6	91,5
Helpterberg	90,5	103,8
Marlow	91,0	100,8
Sonneberg	91,7	102,7
Dresden	92,2	102,4
Inselsberg	92,5	102,2
Schwerin	92,8	101,3
K.-M.-Stadt	92,8	100,0
Suhl	93,7	-
Leipzig	93,9	102,9
Brocken	94,6	101,4
Dequede	94,9	101,0
Löbau	98,2	91,8
Cottbus	98,6	103,2
Berlin	99,7	102,6
Hoyerswerda	100,4	-
Marlow	102,8	-

sowie auf MW :

Burg	-	657 kHz
Neubrandenburg	-	657 kHz
Reichenbach	-	657 kHz

Dieses Heft entstand in Zusammenarbeit von Radio DDR II mit Prof. Dr. Horst Völz und der Abteilung Öffentlichkeitsarbeit des Rundfunks der DDR.

RUNDFUNK DER DDR

Ag 142/58/89